```cpp
/*
TinyTrackGPS.cpp - A simple track GPS to SD card logger.
TinyTrackGPS v0.7

Copyright © 2019-2021 Francisco Rafael Reyes Carmona.
All rights reserved.

rafael.reyes.carmona@gmail.com

  This file is part of TinyTrackGPS.

  TinyTrackGPS is free software: you can redistribute it and/or modify
  it under the terms of the GNU General Public License as published by
  the Free Software Foundation, either version 3 of the License, or
  (at your option) any later version.

  TinyTrackGPS is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
  GNU General Public License for more details.

  You should have received a copy of the GNU General Public License
  along with TinyTrackGPS.  If not, see <https://www.gnu.org/licenses/>.
*/

/*****************************************************************************
/  Programa de localizacion por gps que graba las posiciones en
/  un fichero de texto cada segundo, de forma diaria.
/
/  - Conectar módulo SD con pin CS (naranja) en pin 10 arduino.
/
/  Uso de librería TinyGPS.
/   Requiere uso de librería SoftwareSerial, se presupone que disponemos
/   de un dispositivo GPS serie de 9600-bauds conectado en pines 9(rx) y 8(tx).
/  - Conectar módulo NMEA-6M (gps) pines 8,9 (9 - pin rx negro)
/
/  - Conectar LCD 16x2 pines 2,3,4,5,6,7 (2-amarillo , 3-azul,
/     4-rojo, 5-azul oscuro, 6-verde, 7-blanco)
/
/  - Conectar OLED 0.96" en SDA y SCL. pines A4 y A5 del Arduino UNO.
********************************************************************************/

// Include libraries.
#include <Arduino.h>
#include "config.h"
#include "Display.h"
#include <SoftwareSerial.h>
#include <TinyGPS.h>
#include <SdFat.h>
#include <sdios.h>
#include <LowPower.h>
#include "UTMconversion.h"

// Variables para grabar en SD.
char GPSLogFile[] = "YYYYMMDD.csv"; // Formato de nombre de fichero. YYYY-Año, MM-Mes, DD-Día.

const uint8_t CHIP_SELECT = SS;  // SD card chip select pin. (10)
SdFat card;   //SdFat.h library.
SdFile file;
```

```
boolean SDReady;
boolean SaveOK;

// Variables y clases para obtener datos del GPS y conversion UTM.
TinyGPS gps;
GPS_UTM utm;
SoftwareSerial gps_serial(9, 8);
int year_actual;
byte month_actual, day_actual;
byte hour_prev, minute_prev, second_prev;
float flat, flon;
int year;
byte month, day, hour, minute, second, hundredths;
unsigned long age;
unsigned int elev;

// Definimos el Display
#if defined(DISPLAY_TYPE_LCD_16X2)
Display LCD(LCD_16X2);
#elif defined(DISPLAY_TYPE_LCD_16X2_I2C)
Display LCD(LCD_16X2_I2C);
#elif defined(DISPLAY_TYPE_SDD1306_128X64)
Display LCD(SDD1306_128X64);
#else
#define NO_DISPLAY
#endif

//------------------------------------------------------------------------------
/*
 * User provided date time callback function.
 * See SdFile::dateTimeCallback() for usage.
 */
void dateTime(uint16_t* date, uint16_t* time) {
  // User gets date and time from GPS or real-time
  // clock in real callback function

  // return date using FAT_DATE macro to format fields
  //*date = FAT_DATE(year, month, day);
  *date = (year-1980) << 9 | month << 5 | day;

  // return time using FAT_TIME macro to format fields
  //*time = FAT_TIME(hour, minute, second);
  *time = hour << 11 | minute << 5 | second >> 1;
}
//------------------------------------------------------------------------------

void GPSData(TinyGPS &gps, GPS_UTM &utm);
#ifndef NO_DISPLAY
void ScreenPrint(Display &LCD, TinyGPS &gps, GPS_UTM &utm);
bool pinswitch();
#endif
void GPSRefresh();

unsigned long iteration = 0;

void setup(void) {
  //Serial.begin(9600);
  gps_serial.begin(9600);

  //Serial.print(F("Initializing SD card..."));

  SDReady = card.begin(CHIP_SELECT);
  //(SDReady) ? Serial.println(F("Done.")) : Serial.println(F("FAILED!"));

  /* Iniciaización del display LCD u OLED */
  #ifndef NO_DISPLAY
  LCD.start();
  LCD.clr();
  LCD.splash(750);      // Dibujamos la presensación.
  #endif

  //Serial.print(F("Waiting for GPS signal..."));
  #ifndef NO_DISPLAY
  LCD.clr();
  LCD.print("Waiting for","GPS signal...","");
  unsigned int time = 0;
  #endif

  bool config = false;

  do {
    #ifndef NO_DISPLAY
    LCD.wait_anin(time++);
    #endif
    for (unsigned long start = millis(); millis() - start < 1000;) {
      while (gps_serial.available() > 0) {
        char c = gps_serial.read();
        //Serial.write(c); // uncomment this line if you want to see the GPS data flowing
        if (gps.encode(c)) {// Did a new valid sentence come in?
          gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);
          (age != TinyGPS::GPS_INVALID_AGE) ? config = true : config = false;
        }
      }
    }
  }while(!config);

  sprintf(GPSLogFile, "%04d%02d%02d.csv", year, month, day);

  year_actual = year;
  month_actual = month;
  day_actual = day;
  hour_prev = hour;
  minute_prev = minute;
  second_prev = second;

  //Serial.println(F("Done."));
  //Serial.println(F("Configuration ended."));

  #ifndef NO_DISPLAY
  LCD.clr();
  #endif
}

void loop(void) {
  bool gps_ok = false;

  while (gps_serial.available() > 0) {
    char c = gps_serial.read();
    //Serial.write(c); // uncomment this line if you want to see the GPS data flowing
    if (gps.encode(c)) {
      gps_ok = true;
    }
  }

  gps.f_get_position(&flat, &flon, &age);
  utm.UTM(flat, flon);
  gps.crack_datetime(&year, &month, &day, &hour, &minute, &second, &hundredths, &age);
  if ((elev = gps.altitude()) == TinyGPS::GPS_INVALID_ALTITUDE) elev = 0;
  else elev /= 100L;

  if (gps_ok) {
    GPSRefresh();
    if (!((hour_prev == hour) && (minute_prev == minute) && (second_prev == second))) {
      GPSData(gps, utm);
      iteration++;
    }
    #ifndef NO_DISPLAY
    ScreenPrint(LCD, gps, utm);
    #endif
  }

  LowPower.idle(SLEEP_120MS, ADC_OFF, TIMER2_OFF, TIMER1_OFF, TIMER0_OFF, SPI_ON, USART0_ON, TWI_ON);
}

void GPSData(TinyGPS &gps, GPS_UTM &utm) {
  char buffer[60];
  char line[11];
  int index;
  int zone;
  char band;
  long X;
  long Y;

  zone = utm.zone();
  band = utm.band();
  X = utm.X();
  Y = utm.Y();

  if (age != TinyGPS::GPS_INVALID_AGE){
    index = snprintf(buffer,10, "%02d:%02d:%02d,", hour, minute, second);
    dtostrf(flat, 10, 6, line);
    index += snprintf(buffer+index,12,"%s,",line);
    dtostrf(flon, 10, 6, line);
    index += snprintf(buffer+index,12,"%s,",line);
    index += snprintf(buffer+index,7,"%05u,",elev);
    index += snprintf(buffer+index,19,"%02d%c %ld %ld", zone, band, X, Y);
    //Serial.print(buffer);
  }

  if (year != year_actual || month != month_actual || day != day_actual) {
    sprintf(GPSLogFile, "%04d%02d%02d.csv", year, month, day);
    year_actual = year;
    month_actual = month;
    day_actual = day;
  }

  SdFile::dateTimeCallback(dateTime);

  // Si no existe el fichero lo crea y añade las cabeceras.
  if (SDReady && !card.exists(GPSLogFile)) {
    if (file.open(GPSLogFile, O_CREAT | O_APPEND | O_WRITE)) {
      //Serial.print(F("New GPSLogFile, adding heads..."));
      file.println(F("Time, Latitude, Longitude, Elevation, UTM Coords (WGS84)"));
      //Serial.println(F("Done."));
      file.close();
      }
      //else {
      //Serial.println(F("** Error creating GPSLogFile. **"));
      //}
  }

  //if (!((hour_prev == hour) && (minute_prev == minute) && (second_prev == second))) {
  if (SDReady && file.open(GPSLogFile, O_APPEND | O_WRITE)) {
    //Serial.print(F("Open GPSLogFile to write..."));
    SaveOK = true;
    file.println(buffer);
    file.close();
    //Serial.println(F("Done."));
    hour_prev = hour;
    minute_prev = minute;
    second_prev = second;
  } else {
    SaveOK = false;
    //Serial.println(F("** Error opening GPSLogFile. **"));
    }
  //} //else Serial.println(F("** GPS signal lost. **"));
}

#ifndef NO_DISPLAY
void ScreenPrint(Display &LCD, TinyGPS &gps, GPS_UTM &utm){
  bool print_utm = false;
  bool print_grades = false;
  unsigned short sats;

  sats = gps.satellites();

  if (LCD.display_type() == SDD1306_128X64) {
    print_utm = true;
    print_grades = true;
  }
  else if (!pinswitch()) print_utm = true;
  else print_grades = true;

  if (print_utm) {
    char line[12];

    sprintf(line, "%02d%c %ld ", utm.zone(), utm.band(), utm.X());
    //Serial.println(line);
    LCD.print(0,0,line);
    LCD.print_PChar((byte)6);
    sprintf(line, "%02hu ", sats);
    //Serial.println(line);
    LCD.print(12,0,line);
    SaveOK ? LCD.print_PChar((byte)7) : LCD.print("-");

    // New line
    sprintf(line, "%ld ", utm.Y());
    //Serial.println(line);
    LCD.print(1,1,line);
    LCD.print_PChar((byte)5);
    LCD.print(10,1,"_____");
    sprintf(line, "%um", elev);
    //Serial.println(line);

    if (elev < 10) LCD.print(14,1,line);
    else if (elev < 100) LCD.print(13,1,line);
    else if (elev < 1000) LCD.print(12,1,line);
    else LCD.print(11,1,line);
  }

  if (print_grades) {
    char line[11];
    LCD.print(1,(LCD.display_type() == SDD1306_128X64) ? 2 : 0,"LAT=");
    dtostrf(flat, 10, 6, line);
    LCD.print(line);
    LCD.print(1,(LCD.display_type() == SDD1306_128X64) ? 3 : 1,"LON=");
    dtostrf(flon, 10, 6, line);
    LCD.print(line);
  }
}

bool pinswitch()
{
  bool pin;

  if (LCD.display_type() == SDD1306_128X64) return true;

  pin = bitRead(iteration,4); // Change every 8 seconds.
  //LCD.clr(); -> Too slow clear individual characters.
  if ((iteration%16) == 0) {
    LCD.print(0,0," ");
    LCD.print(15,0," ");
    LCD.print(0,1," ");
    LCD.print(15,1," ");
  }
  return pin;
}
#endif

void GPSRefresh()
{
    while (gps_serial.available() > 0)
      gps.encode(gps_serial.read());
}

```